lab5 report

于新雨 计25 2022010841

实现功能

按照文档上算法实现死锁检测功能
具体来说，我们对于每个 ProcessControlBlock 维护该算法中的 Available、Allocation、Need 三个矩阵，并且对于每个 process 维护是否进行死锁检测
我们在 `sys_semaphore_create` `sys_mutex_lock` `sys_semaphore_up` 这些关于 mutex 和 semaphore 操作的系统调用中，维护这些矩阵的值
在 `sys_mutex_lock` 和 `sys_semaphore_down` 中，我们会检查当前的矩阵是否满足安全性条件，如果不满足，则说明发生了死锁，返回错误

问答题

1.

分为两类：进程级资源和线程级资源
需要回收的进程级资源（ProcessControlBlock层面）：

- 内存资源：memory_set（地址空间、页表等）
- 文件资源：fd_table中的所有打开文件描述符
- 同步原语资源：
  - mutex_list 中的互斥锁
  - semaphore_list 中的信号量
  - condvar_list 中的条件变量
- 线程管理资源：task_res_allocator（线程资源分配器）

线程级资源（TaskControlBlock 层面）：

- 内核栈：每个线程的kstack
- 用户资源：TaskUserRes（如用户栈、线程ID等）
- 陷入上下文：trap_cx_ppn对应的物理页面

2.

区别：

- mutex1 加锁时使用loop循环重试，mutex2 只检查一次是否可以加锁
- mutex1 释放锁时直接设置locked = false， mutex2 只有在没有等待线程时才设置locked = false

问题：mutex2 可能导致线程对锁状态判断有误，它只有在wait_queue为空时才设置locked = false，当有等待线程时，锁保持locked = true状态，这导致被唤醒的线程无法区分锁是否真正可用，从而可能在之前线程释放锁的时候继续等待